Combinaisons de boules de mots pour la classification de séquences

Frédéric Tantini, Alain Terlutte, Fabien Torre

Combinaisons de boules de mots pour la classification de séquences

Frédéric Tantini[1], Alain Terlutte[2], Fabien Torre[2]

[1] Parole, LORIA UMR 7503 Nancy

[2] Mostrare (INRIA Lille Nord Europe et CNRS LIFL)
Université Lille Nord de France

Résumé : Nous nous intéressons à la combinaison de méthodes d'apprentissage à base de moindres généralisés et de techniques d'inférence de boules de mots, appliquée à la classification de séquences. Nous montrons que cette intégration n'est pas directe et qu'elle nécessite un travail de compréhension des moindres généralisés et des algorithmes nouveaux pour produire des boules de mots.

L'apprenant résultant de cette démarche est ensuite évalué expérimentalement sur des données classiques de l'inférence grammaticale et sur un jeu de données réel. Dans les deux cas, notre approche se montre compétitive avec les meilleurs algorithmes connus.

Mots-clés : moindres généralisés, boules de mots, méthodes d'ensemble, inférence grammaticale, classification de séquences, reconnaissance de caractères.

Introduction

Nous nous intéressons à la classification de séquences, à travers des méthodes fondées en *inférence grammaticale*. Même si la tâche de classification diffère de l'identification de langages, nous pensons que les méthodes d'inférence grammaticale peuvent enrichir des techniques d'apprentissage supervisé.

Cet article relate nos efforts pour combiner deux techniques d'apprentissage :

– les méthodes à base de moindres généralisés ;
– l'inférence de boules de mots.

La première est un cadre général pour la classification supervisé théorisé depuis (Ganascia, 1993) et dont la pertinence a été montrée, entre autres, pour les représentations attributs-valeurs (Torre, 2004), et pour la classification de séquences par automates (Torre & Terlutte, 2009). La seconde est issue de l'inférence grammaticale et bénéficie dans ce cadre de résultats d'apprenabilité (Tantini, 2009).

À travers la volonté de combiner ces deux techniques, deux objectifs sont poursuivis :

– définir une ligne de conduite dans le cas où l'on souhaite mettre en œuvre une technique d'apprentissage à base de moindres généralisés mais où l'espace d'hypothèses choisi ne vérifie les prérequis théoriques ;
– établir l'intérêt des boules pour des problèmes réels de classification de séquences.

Le plan est le suivant. Section 1, nous rappelons les bases d'un apprentissage à base de moindres généralisés et nous mettons en avant les propriétés de cet apprentissage qui pour beaucoup reposent sur l'*unicité du moindre généralisé*. Nous nous plaçons ensuite, section 2, dans un cadre où le moindre généralisé d'un ensemble d'exemples n'est pas unique, empêchant ainsi l'utilisation directe des algorithmes de la section 1. Ce cadre est celui des boules d'exemples. Nous proposons alors des calculs de boules approchant au mieux les propriétés du moindre généralisé unique. À la section 3, nous mettons en œuvre ces algorithmes sur des problèmes réels de classification de séquences, comme par exemple la reconnaissance de caractères manuscrits. Enfin, section 4, nous faisons un bilan de ce travail et en proposons les perspectives.

1 Apprentissage par moindres généralisés

Nous nous inscrivons ici dans la méthodologie d'apprentissage proposée dans (Torre, 2004; Torre & Terlutte, 2009) et que nous généralisons sous la forme suivante :

1. choisir des langages de représentations pour les exemples ($\mathcal{E}$) et pour les hypothèses ($\mathcal{H}$) tels que $\mathcal{E} \subset \mathcal{H}$;
2. se donner une relation de subsomption $\succeq$ sur $\mathcal{H} \times \mathcal{H}$ permettant de vérifier qu'une hypothèse *couvre* un exemple, mais aussi de déterminer si une hypothèse est plus générale qu'une autre (c'est-à-dire, si les exemples subsumés par l'une contient les exemples subsumés par l'autre) ;
3. définir un opérateur G qui, étant donné une hypothèse et un exemple, propose une généralisation de l'hypothèse subsumant l'exemple.

Dans les travaux publiés, l'opérateur G est toujours un calcul de *moindre-généralisé unique* et le point 3 se décline alors de la manière suivante :
– montrer que, pour un ensemble d'exemples quelconque, $\mathcal{H}$ ordonné par $\succeq$ ne permet l'existence que d'une unique hypothèse moindre généralisée de cet ensemble ;
– écrire un algorithme qui calcule cette hypothèse moindre généralisée à partir d'une hypothèse et d'un exemple (G est alors noté MG).

Sous ces contraintes de moindre-généralisé unique et calculable, le système VOLATA [1] fournit des algorithmes complets et génériques (c'est-à-dire indépendants des choix de langages et de subsomption), selon les objectifs visés : DLG pour un apprentissage rapide, GLOBO pour l'obtention d'une théorie compréhensible, les méthodes d'ensemble GLOBOOST et ADABOOST-MG pour de meilleures performances en prédiction.

La suite de cette section vise à dégager les caractéristiques de cette architecture. Pour cela, nous détaillons une partie des algorithmes génériques de haut niveau, puis nous présentons certaines propriétés théoriques qui découlent du moindre généralisé unique.

1.1 Algorithmes GLOBOOST et GC

GLOBOOST est décrit à l'algorithme 1 et vise à produire aléatoirement T hypothèses correctes qui constitueront un classifieur en votant avec un poids identique. À chacune

1. http://www.grappa.univ-lille3.fr/~torre/Recherche/Softwares/volata/

des T étapes, une classe est choisie au hasard, les ensembles d'exemples et de contre-exemples sont constitués, puis l'algorithme GC (pour *généralisation correcte*) est appelé sur ces exemples mélangés aléatoirement.

Algorithm 1 GLOBOOST

Require: des exemples étiquetés (x_i, y_i) et T un nombre d'itérations.
Ensure: H le classifieur final.

for $t = 1$ to T **do**
    target = classe choisie au hasard
    $P = [x_i | y_i = \textit{target}]$
    $N = [x_i | y_i \neq \textit{target}]$
    mélanger P aléatoirement
    $h_t = \text{GC}(P,N)$ /* Appel à l'algorithme GC */
end for
return $H(x) = \text{sign}\left(\sum_{t=1}^{T} h_t(x)\right)$

GC a pour rôle de produire une hypothèse correcte, avec une dépendance à l'ordre des exemples pour favoriser la diversité de l'ensemble d'hypothèses appris. Il est décrit à l'algorithme 2 et à nouveau le principe est simple : le premier exemple sert d'hypothèse-graine avec laquelle on essaye de généraliser, à l'aide de l'opérateur G, chaque autre exemple positif ; chaque étape de généralisation doit être validée vis-à-vis des contre-exemples, un exemple qui amène une généralisation incorrecte n'est pas accepté, on continue alors l'ajout de positifs à partir de la dernière généralisation correcte.

Algorithm 2 GC

Require: $E = [p_1, \ldots, p_n] \subseteq \mathcal{E}$ un ensemble *ordonné* de n exemples de même classe, $N \subseteq \mathcal{E}$ un ensemble de contre exemples.
Ensure: $h \in \mathcal{H}$ généralisant en partie E et correcte vis-à-vis de N.

$g = p_1$
for $i = 2$ to n **do**
    $g' = \text{G}(g, p_i)$ /* généralisation visant à intégrer p_i */
    if $(\forall e \in N : g' \not\succeq e)$ **then** /* si g' est correcte */
        $g = g'$ /* g' devient la généralisation courante */
    end if
end for
return $h(x) = \text{class}(E)$ si $g \succeq x$, 0 sinon (abstention)

Les deux algorithmes présentés, GLOBOOST et GC, sont génériques : les choix de langages et de subsomption apparaîtront uniquement dans G dont le rôle est de généraliser une hypothèse avec un exemple. C'est cet algorithme que nous discutons dans la suite.

1.2 Choix de G et propriétés

Comme indiqué au début de cette section, le candidat privilégié pour le rôle de G, systématiquement utilisé jusque là, est un calcul de moindre généralisé unique.

Définition 1 (moindre généralisé)
Étant donné un ensemble d'exemples $E \subseteq \mathcal{E}$, une hypothèse $h \in \mathcal{H}$ est dite moindre généralisée *de E si et seulement si :*
– $\forall e \in E : h \succeq e$;
– *il n'existe pas h' vérifiant* $\forall e \in E : h' \succeq e$ *et* $h \succ h'$.

Monotonie de l'opérateur G.

Si G calcule un moindre généralisé ainsi défini (on peut considérer sans perte de généralité G sous sa forme incrémentale), alors GC recherche une hypothèse de manière ascendante (du spécifique au général) et les hypothèses produites successivement au cours d'une exécution de GC sont incluses les unes dans les autres. Cette monotonie de G garantit qu'un exemple couvert à un instant le sera aussi par les hypothèses suivantes. Nous avons alors GC qui suit un chemin dans $\mathcal{H}$ en procédant par plus petit pas à l'aide de G : il n'est pas possible d'être plus prudent dans l'espace de recherche choisi.

Existence d'un moindre généralisé unique.

Il faut noter que l'utilisation que nous faisons de G dans GC et donc dans GLOBOOST, sous-entend que pour toute hypothèse $h \in \mathcal{H}$ et tout exemple $e \in \mathcal{E}$, il existe une *unique* hypothèse qui soit moindre généralisée de h et de e et que G calcule cette hypothèse. Cela met en avant que la définition 1 n'implique pas l'unicité du moindre-généralisé, nous le vérifierons à la section 2 avec un couple $(\mathcal{H}, \succeq)$ qui induit des moindres généralisés multiples. Lorsque cette unicité est garantie, nous notons MG l'opérateur correspondant. On voit bien cette fois que celui-ci est fortement lié aux choix de $\mathcal{E}$, $\mathcal{H}$ et $\succeq$. Surtout, il nous assure une indéniable efficacité : la séquence d'exemples à généraliser n'induit qu'un unique chemin dans $\mathcal{H}$. En plus d'avancer par plus petits pas, GC n'a pas de point de choix à gérer, ni de *backtrack* à effectuer, ni d'ensemble d'hypothèses candidates à maintenir.

Monotonie de GC *et maximalité des hypothèses apprises par* GC.

GC dotée de MG présente une autre forme de monotonie : un exemple positif rejeté par GC ne sera couvert par aucune hypothèse produite ensuite au cours du même GC. Autrement dit, il n'existe pas d'hypothèse dans $\mathcal{H}$ qui subsume à la fois l'hypothèse produite par GC et un exemple de l'échantillon rejeté par GC, sans couvrir de négatif : l'hypothèse produite par GC est en ce sens *maximalement* correcte.

Granularité induite par G.

Nous venons de voir le cas idéal pour une méthode comme GLOBOOST, c'est-à-dire quand $(\mathcal{H}, \succeq)$ définit un moindre généralisé unique calculable par MG. Cette condition revient à demander que $\mathcal{H}$ dotée de $\succeq$ soit un sup-demi-treillis. Cela nous amène à revenir aux choix de $\mathcal{H}$ et de $\succeq$. Il apparaît clairement qu'ils ont un rôle crucial puisqu'ils définissent la *granularité* de l'espace de recherche :
– si le maillage est trop fin, l'apprentissage par cœur est possible et MG ne généralise pas du tout ; GLOBOOST produira une hypothèse par classe reconnaissant unique-

ment les exemples de l'échantillon d'apprentissage et incapable de classer de nouveaux exemples ;
– si le maillage est trop grossier, toute généralisation d'exemples positifs par MG couvre des négatifs et GC va alors produire une hypothèse par exemple ; à nouveau il s'agit d'un par cœur incapable de prédiction.

Récapitulatif des propriétés d'un apprentissage idéal à l'aide de GLOBOOST.

– G avance par plus petits pas ;
– G est monotone ;
– GC produit des hypothèses correctes ;
– GC est dépendant à l'ordre de présentation des positifs ;
– GC est monotone, GC produit des hypothèses maximales ;
– $(\mathcal{H}, \succeq)$ a une granularité qui présente un bon compromis.

Si nous ne sommes pas dans cette situation idéale, c'est typiquement le cas lorsqu'il n'y a pas unicité du moindre généralisé, il faut chercher un algorithme de substitution tout en sachant que les propriétés idéales de G et GC ne peuvent être réunies toutes en même temps que par l'utilisation de MG. Il s'agit donc, malgré tout, de préserver un maximum de ces propriétés. C'est la démarche que nous développons à la section suivante, pour le cas où les hypothèses sont des boules d'exemples.

2 Boules de mots

2.1 Définitions

Nous nous plaçons maintenant dans un cas concret proche de l'inférence grammaticale : les exemples $\mathcal{E}$ sont les mots construits sur un alphabet Σ fixé : $\mathcal{E} = \Sigma^*$. Les hypothèses considérées sont des *boules de mots*, une boule étant définie par un mot-centre o, un rayon r et étant notée $B_r(o)$. Une boule de mots $B_r(o)$ *contient* l'ensemble des mots à distance inférieure ou égale à r de o, c'est-à-dire, $B_r(o) = \{w \in \Sigma^* | d(o, w) \leq r\}$. Autrement dit, le test de subsomption $\succeq$ entre une hypothèse $h = B_r(o)$ et un exemple e est simplement le test d'appartenance d'un mot à la boule :

$$h \succeq e \Leftrightarrow d(e, o) \leq r$$

La distance utilisée est la distance d'édition, ou distance de Levenshtein (1965), pour laquelle chaque opération d'édition parmi (insertion, suppression, substitution) a un coût unitaire. On peut la définir comme étant la longueur minimale des dérivations de réécriture permettant de passer d'un premier mot à un second. Plus précisément, étant donnés deux mots $w, w' \in \Sigma^*$, on dit que w se réécrit en w' en un pas, noté $w \to w'$ si l'une des conditions suivantes est vérifiée :

– *opération de suppression* : $w = uav$ et $w' = uv$ avec $u, v \in \Sigma^*$ et $a \in \Sigma$;
– *opération d'insertion* : $w = uv$ et $w' = uav$ avec $u, v \in \Sigma^*$ et $a \in \Sigma$;
– *opération de substitution* : $w = uav$ et $w' = ubv$ avec $u, v \in \Sigma^*$, $a, b \in \Sigma$, $a \neq b$.

Nous noterons $w \xrightarrow{k} w'$ si w peut se réécrire en w' à l'aide de k opérations d'édition.

Définition 2 (Distance d'édition)
La distance d'édition entre deux mots w et w', notée $d(w, w')$, est le nombre minimum d'opérations d'édition nécessaires pour réécrire w en w' ; c'est donc le plus petit entier $k \in \mathbb{N}$ tel que $w \xrightarrow{k} w'$.

Exemple 1
On peut montrer que $d(abaa, aab) = 2$ car $abaa$ nécessite au moins deux pas pour être transformé en aab : $a\underline{b}aa \rightarrow aa\underline{a} \rightarrow aab$. Notons que ce n'est pas la seule façon de passer du mot $abaa$ au mot aab : en effectuant d'abord la substitution du a par un b, la transformation devient $aba\underline{a} \rightarrow a\underline{b}ab \rightarrow aab$.

Le calcul de la distance $d(w, w')$ est généralement effectué en remarquant que :

$$d(ua, vb) = min \begin{cases} d(ua, v) & + & 1 \\ d(u, v) & + & \text{coût(substitution)} \\ d(u, vb) & + & 1 \end{cases}$$

où $\text{cout(substitution)}$ vaut 0 si $a = b$, 1 sinon. L'algorithme 3 effectue ce calcul en $\mathcal{O}(|w| \cdot |w'|)$ par programmation dynamique (Wagner & Fischer, 1974). Son but est de remplir une matrice M de telle façon que $M\left[|w|+1\right]\left[|w'|+1\right]$ contienne la distance d'édition $d(w, w')$, en calculant la distance d'édition entre chaque préfixe de w et w'.

Algorithm 3 Algorithme général du calcul de la distance d'édition

Require: Deux mots w et w'
Ensure: $d(w, w') = M\left[|w|+1\right]\left[|w'|+1\right]$
1: $M[0][0] = 0$
2: **for** $i = 0$ to $|w|$ **do**
3: $\quad M[i+1][0] = M[i][0] + 1$ /* Initialisation de la colonne */
4: **end for**
5: **for** $i = 0$ to $|w'|$ **do**
6: $\quad M[0][i+1] = M[0][i] + 1$ /* Initialisation de la ligne */
7: **end for**
8: **for** $i = 0$ to $|w|$ **do**
9: $\quad$ **for** $j = 0$ to $|w'|$ **do**
10: $\quad\quad M[i+1][j+1] = \min \begin{cases} M[i+1][j] + 1 \\ M[i][j] + \text{cout(substitution)} \\ M[i][j+1] + 1 \end{cases}$
11: $\quad$ **end for**
12: **end for**

Il est intéressant de noter qu'en gardant en mémoire un pointeur sur la case d'où provient le résultat de $M[i][j]$, on peut retrouver les opérations d'édition nécessaires pour passer d'un mot à l'autre.

Exemple 2
Le calcul de la distance d'édition entre w = ABACD et w' = EAFCGD par l'algorithme 3 permet de remplir M de la manière suivante :

		E	*A*	*F*	*C*	*G*	*D*
	0	*1*	*2*	*3*	*4*	*5*	*6*
A	*1*	*1*	*1*	*2*	*3*	*4*	*5*
B	*2*	*2*	*2*	*2*	*3*	*4*	*5*
A	*3*	*3*	*2*	*3*	*3*	*4*	*5*
C	*4*	*4*	*3*	*3*	*3*	*4*	*5*
D	*5*	*5*	*4*	*4*	*4*	*4*	*4*

On en déduit que la distance entre les deux mots est de $d(ABACD, EAFCGD) = 4$. *De plus, si l'on a mémorisé les étapes de calcul, on peut retrouver les différents chemins permettant de passer d'un mot à l'autre. On peut par exemple aligner les mots de la façon suivante :*

$$\begin{matrix} - & A & B & A & C & D \\ E & A & F & C & G & D \end{matrix}$$

Les opérations nécessaires pour passer de ABACD à EAFCGD sont donc l'insertion d'un E et les substitutions d'un B par un F, d'un A par un C, et d'un C par un G. Toutefois, on trouve que l'alignement suivant est lui aussi possible :

$$\begin{matrix} - & A & B & A & C & - & D \\ E & A & F & - & C & G & D \end{matrix}$$

Les opérations de réécriture sont alors différentes : il faut insérer un E et un G, supprimer un A et substituer un B par un F.

2.2 Généralisations de mots en boules

Les exemples étant des mots et les hypothèses des boules de mots, nous nous intéressons aux opérateurs G généralisant une boule et un mot en une nouvelle boule.

(Non-)existence d'un moindre généralisé unique.

D'emblée, nous pouvons noter que dans cet espace les hypothèses moindres généralisées sont multiples.

Exemple 3
Soient $E = [a, b, ab]$, $h = B_1(a)$ *et* $h' = B_1(b)$. *Les deux hypothèses subsument bien les trois exemples (*$h \succeq E$ *et* $h' \succeq E$*) mais* $h' \not\succeq h$ *et* $h \not\succeq h'$.

Cela est intuitif dans le plan : la figure 1 exhibe trois points du plan et plusieurs hypothèses-cercles couvrant ces points mais incomparables entre elles. Elle nous convainc ainsi que nous pouvons avoir une infinité d'hypothèses moindres généralisées.

En d'autres termes, nous ne sommes manifestement pas dans le cadre idéal évoqué à la section 1 dans lequel on aurait l'unicité du moindre généralisé.

Certains pourraient cependant souligner l'existence d'une *plus petite boule* contenant les exemples à généraliser, comme cela est visible sur la figure 1. C'est l'occasion pour nous de mettre en avant que cette notion de *plus petite boule* ne se confond pas avec celle de *moindre généralisé* : aussi petite soit elle, cette hypothèse n'est pas contenue dans les autres moindres généralisés. Autrement dit, il n'y a pas de raison particulière pour

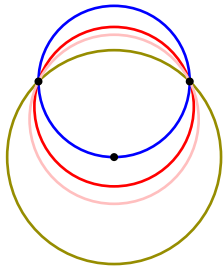

FIGURE 1 – Infinité de cercles contenant trois points et incomparables entre eux.

que cette *plus petite boule* soit, plus qu'une autre, l'amorce du concept cible. De plus, dans le cadre qui nous intéresse, cette piste est bloquée par la complexité du calcul de la boule de rayon minimum. En effet, le théorème suivant permet de prouver que trouver le centre de la plus petite boule contenant un ensemble fini de mots est NP-difficile.

Théorème 4 (de la Higuera & Casacuberta (2000))
Étant donnés un ensemble fini de mots $W = \{w_1, \ldots, w_n\}$ et une constante K, le problème consistant à décider si un mot $z \in \Sigma^*$ existe tel que $\sum_{w \in W} d(z, w) < K$ (respectivement $\max_{w \in W} d(z, w) < K$) est NP-complet.

Proposition d'un opérateur G *monotone.*

Pour palier à ces problèmes, nous proposons l'algorithme incrémental 4 en guise d'opérateur G.

Algorithm 4 Algorithme générique G du calcul de la boule généralisée

Require: $e \in \mathcal{E}$ un exemple, $h = B_r(o) \in \mathcal{H}$ une hypothèse.
Ensure: $g \in \mathcal{H}$ une généralisation de e et h ($g \succeq h$ et $g \succeq e$).

```
c = o →* e /* chemin de longueur minimale */
Soit u un mot sur le chemin c /* c = o →x u →y e, x + y = d(o, e) */
x = d(o, u)
y = d(u, e)
k = max(x + r, y)
return B_k(u)
```

Cet algorithme doit être instancié par la méthode choisissant le nouveau centre u sur le chemin c mais, quel que soit ce choix, la nouvelle boule produite a une propriété de monotonie qui nous intéresse, c'est-à-dire qu'elle inclut bien l'hypothèse précédente et le nouvel exemple :

- $g \succeq e$, en effet $d(u, e) = y \leq k$. On a donc bien $e \in B_k(u)$;
- $g \succeq h$, en effet, pour tout mot $w \in B_r(o)$ on a $d(o, w) \leq r$; de plus, par l'inégalité triangulaire, $d(u, w) \leq d(u, o) + d(o, w)$ et ainsi $d(u, w) \leq x + r \leq k$; donc $w \in B_k(u)$.

Comme son exécution repose principalement sur le calcul du chemin entre le centre et le nouvel exemple, la complexité de cet algorithme est la même que celle de l'algorithme 3, c'est-à-dire polynomiale en la longueur des mots. Malheureusement, la contrepartie de la monotonie et de ce gain en complexité est que la boule calculée n'est pas nécessairement une hypothèse moindre généralisée.

Exemple 5
Soit $E = [a, b]$. La première hypothèse est le premier exemple, c'est-à-dire $h = B_0(a)$. Ensuite, comme le chemin entre le centre et le nouvel exemple b vaut $c : a \xrightarrow{1} b$, il existe deux possibilités de choisir u. Ainsi, soit $\text{G}(h, b) = B_1(a)$, soit elle vaut $\text{G}(h, b) = B_1(b)$. Or la boule unité centrée sur le mot vide $B_1(\lambda)$ contient bien E et est plus spécifique que les hypothèses calculées : $B_1(\lambda) \subseteq B_1(a)$ et $B_1(\lambda) \subseteq B_1(b)$.

Cependant, la complexité combinatoire des boules de mots nous empêche de fournir une meilleure construction.

Monotonie de GC *et maximalité des hypothèses apprises par* GC.

Nous n'avons pas non plus la monotonie de GC.

Exemple 6
En effet, si l'on adopte par exemple la stratégie qui consiste à choisir le nouveau centre à une distance de 1 de l'ancien centre ($x = 1$), si les exemples sont λ, b, a et si le seul contre-exemple est bb, alors les hypothèses successivement produites sont :
- *$B_0(\lambda)$ est l'hypothèse initiale ;*
- *$B_1(b)$ est rejetée car elle couvre bb ;*
- *$B_1(a)$ est acceptée.*

Cette dernière couvre b alors que l'ajout de ce mot a été rejeté à l'étape précédente.

Granularité induite par G.

Il est difficile d'évaluer la granularité de notre opérateur, par contre le résultat suivant sur la VC-dimension des boules peut alimenter notre réflexion.

Théorème 7 (Janodet (2010))
La VC-dimension des boules, sur un alphabet à deux lettres, est infinie.

Preuve : Nous prenons n mots, tous de longueur n, définis comme suit : le i^{e} mot n'est constitué que de $\texttt{a}$ à l'exception de la i^{e} lettre qui est un $\texttt{b}$. Considérons maintenant que ces mots sont étiquetés : k positivement, $(n - k)$ négativement. On peut construire une boule qui ne capture que les positifs comme suit : le centre est le mot de longueur n qui a des $\texttt{b}$ aux mêmes positions que les positifs et des $\texttt{a}$ ailleurs, le rayon vaut $(k - 1)$. Par construction, les positifs sont accessibles à partir du centre après $(k - 1)$ substitutions, tandis que les négatifs sont à une distance strictement plus grande. □

Dans cette preuve, on peut remarquer que la boule construite contient plus de mots que ceux de l'échantillon (signe qu'il y a eu généralisation et pas apprentissage par

cœur) et que les mots sont placés à la périphérie de la boule d'autre part (indiquant que la boule apprise reste relativement spécifique aux exemples de l'échantillon).

Récapitulatif des propriétés de notre calcul de boules.

- G n'avance pas par plus petits pas (exemple 5) ;
- G est monotone ;
- GC produit des hypothèses correctes ;
- GC est dépendant à l'ordre de présentation des positifs ; il faut noter que les boules ajoutent ici à la dépendance naturelle de GC car même en l'absence de contre-exemples, la boule généralisée change avec l'ordre d'arrivée des exemples ;
- GC n'est pas monotone et ne produit pas nécessairement des hypothèses maximales (exemple 6) ;
- $(\mathcal{H}, \succeq)$ a une granularité différente des cas extrêmes défavorables.

2.3 Calculs du centre de la nouvelle boule

Dans l'algorithme 4, c est un chemin de réécriture de o en e. Or, comme nous l'avons vu aux exemples 1 et 2, il existe souvent plusieurs façons de passer d'un mot à l'autre. Le calcul de ce chemin est effectué en partant de la dernière case de la matrice et en remontant suivant les origines de la valeur de cette case.

Afin de toujours calculer le même chemin $c = o \xrightarrow{*} e$, nous avons effectué les choix arbitraires suivants :

- si le calcul de $M[i][j]$ peut provenir d'une suppression ou d'une autre opération, nous choisissons la suppression ;
- si le calcul de $M[i][j]$ peut provenir d'une insertion ou d'une substitution, nous choisissons l'insertion ;
- une fois les opérations d'édition trouvées, nous effectuons la réécriture de gauche à droite, c'est-à-dire dans le sens de lecture.

Nous appellerons *chemin d'édition* un tel chemin. Fixer de tels choix permet alors à l'algorithme GC d'être dépendant de l'ordre des exemples tout en étant déterministe.

Exemple 8
Soient les mots w = ABACD et w' = EAFCGD. La table 1 contient le calcul des distances d'édition d(ABACD, EAFCGD) et d(EAFCGD, ABACD). Les cases grisées retracent la sélection des opérations d'édition pour le calcul du chemin d'édition telle que défini précédemment.

En appliquant les opérations de gauche à droite dans le mot, on obtient alors les chemins d'édition suivants :

- $\underline{\cdot}ABACD \rightarrow EA\underline{B}ACD \rightarrow EAF\underline{A}CD \rightarrow EAFC\underline{\cdot}D \rightarrow EAFCGD$
- $\underline{E}AFCGD \rightarrow A\underline{\cdot}AFCGD \rightarrow ABA\underline{F}CGD \rightarrow ABAC\underline{G}D \rightarrow ABACD$

Enfin, plusieurs stratégies sont possibles pour le choix du nouveau centre u le long du chemin d'édition. Ce choix peut dépendre du problème traité ou d'heuristiques selon que l'on veuille accorder de l'importance à la graine, favoriser les exemples longs, *etc.*

		E	A	F	C	G	D
	0	1	2	3	4	5	6
A	1	1	1	2	3	4	5
B	2	2	2	2	3	4	5
A	3	3	2	3	3	4	5
C	4	4	3	3	3	4	5
D	5	5	4	4	4	4	4

		A	B	A	C	D
	0	1	2	3	4	5
E	1	1	2	3	4	5
A	2	1	2	2	3	4
F	3	2	2	3	3	4
C	4	3	3	3	3	4
G	5	4	4	4	4	4
D	6	5	5	5	5	4

TABLE 1 – Matrice du calcul des chemins d'édition ABACD $\xrightarrow{*}$ EAFCGD (gauche) et EAFCGD $\xrightarrow{*}$ ABACD (droite).

- NATURELLE : $x = y$ (si k pair, sinon, $x = y + 1$). Version *naïve*, identique à celle qui serait utilisée dans l'espace euclidien, elle "minimise" le rayon de la nouvelle boule.
- GRAINE : pondérée par le nombre d'exemples dans la boule. On donne plus d'importance à la graine.
- CENTRELONG : pondéré par la longueur de l'ex-centre $\left(x = d(o,e) \times \lceil \frac{d(o,e)}{d(o,e)+|o|} \rceil\right)$. Plus le centre est long, plus on se rapproche de lui.
- CENTRECOURT : pondéré par la longueur de l'ex-centre $\left(x = d(o,e) \times \lceil \frac{|o|}{d(o,e)+|o|} \rceil\right)$. Plus le centre est long, plus on s'en éloigne.
- HASARD : le nouveau centre est sur le chemin, au gré du hasard $(x \in [0; d(o,e)])$.

Nous avons également envisagé de produire des boules centrées sur le premier exemple fourni. Dans ce cas, il n'y a plus besoin ni de G ni de GC pour apprendre le rayon de la boule : il suffit de prendre la distance au plus proche contre-exemple. Dans ce cas, nous ne sommes plus guidés par les exemples et perdons énormément en diversité : chaque exemple engendre une unique boule. Ces intuitions sont confirmées expérimentalement par des résultats désastreux pour une telle stratégie.

3 Expérimentations

Parmi les méthodes susceptibles d'intégrer notre algorithme de généralisation de boules, nous conservons GLOBOOST, déjà décrit à l'algorithme 1. Nous avons mis en œuvre le protocole suivant : validation croisée 10 fois et GLOBOOST exécuté 10 fois sur chaque bloc de validation croisée, ce qui signifie qu'un résultat donné pour GLOBOOST correspond à une moyenne sur 100 exécutions.

3.1 Données de l'UCI Repository

Nous utilisons dans cette section les jeux de données séquentiels de l'*UCI Repository* Blake & Merz (1998), à savoir : `tic-tac-toe`, `badges`, `promoters`, `us-first--name` et `splice`. Ces quelques problèmes permettent de considérer des alphabets dont la taille varie de 3 à 30 lettres. Le protocole décrit donne 90 % des données pour

TABLE 2 – Précision des différentes méthodes sur les bases de l'UCI Repository. G-M signifie GLOBOOST instancié par M, où M peut être un calcul d'automates k-testables (TSSI), 0-réversible (ZR), ou un calcul de boules de mots (B). Dans ce dernier cas, la méthode de choix du nouveau centre est précisée en indice.

(références)	tic-tac-toe	badges	promoters	first-name	splice
Majorité	65.34 %	**71.43 %**	50.00 %	81.62 %	50.26 %
RPNI	91.13 %	62.24 %	-	81.42 %	-
TRAXBAR	90.81 %	57.48 %	56.60 %	81.37 %	**58.33 %**
RED-BLUE	**93.89 %**	61.09 %	**63.02 %**	**82.83 %**	54.65 %
G-TSSI	91.47 %	**72.69 %**	**61.13 %**	**89.50 %**	**78.07 %**
G-ZR	**98.36 %**	71.43 %	50.00 %	83.07 %	-
(GLOBOOST $\times$1 000)	tic-tac-toe	badges	promoters	first-name	splice
G-B$_{\text{NATURELLE}}$	92.64 %	81.10 %	**88.58 %**	87.24 %	**93.78 %**
G-B$_{\text{GRAINE}}$	**92.77 %**	**81.72 %**	86.13 %	**87.45 %**	93.63 %
G-B$_{\text{CENTRELONG}}$	91.04 %	81.12 %	86.53 %	86.84 %	93.48 %
G-B$_{\text{CENTRECOURT}}$	74.89 %	80.43 %	87.55 %	86.95 %	92.70 %
G-B$_{\text{HASARD}}$	92.62 %	80.41 %	87.63 %	87.10 %	93.76 %
(GLOBOOST $\times$10 000)	tic-tac-toe	badges	promoters	first-name	splice
G-B$_{\text{NATURELLE}}$	94.54 %	81.38 %	**88.90 %**	88.59 %	95.17 %
G-B$_{\text{GRAINE}}$	94.14 %	82.04 %	86.30 %	88.93 %	95.29 %
G-B$_{\text{CENTRELONG}}$	94.69 %	**82.21 %**	86.75 %	**89.47 %**	95.54 %
G-B$_{\text{CENTRECOURT}}$	74.46 %	81.12 %	88.63 %	89.37 %	**95.83 %**
G-B$_{\text{HASARD}}$	**94.69 %**	81.39 %	88.43 %	88.80 %	95.63 %
(GLOBOOST $\times$100 000)	tic-tac-toe	badges	promoters	first-name	splice
G-B$_{\text{NATURELLE}}$	94.43 %	81.21 %	**89.74 %**	88.72 %	95.92 %
G-B$_{\text{GRAINE}}$	94.26 %	81.39 %	86.58 %	89.11 %	95.37 %
G-B$_{\text{CENTRELONG}}$	**95.03 %**	**81.98 %**	87.58 %	**89.93 %**	95.46 %
G-B$_{\text{CENTRECOURT}}$	74.45 %	81.13 %	89.20 %	89.88 %	**96.12 %**
G-B$_{\text{HASARD}}$	94.96 %	81.36 %	89.08 %	89.06 %	95.62 %

apprendre tandis que les 10 % restants sont gardés pour le test. L'objectif est ici de nous comparer à des méthodes classiques en Inférence Grammaticale (RPNI) et aussi à GLOBOOST instancié par des calculs d'automates moindres généralisés (TSSI et ZR) comme proposé par Torre & Terlutte (2009). Les résultats sont donnés à la table 2 pour 1 000, 10 000 et 100 000 boules produites par GLOBOOST sur chaque problème considéré.

3.2 Reconnaissance de chiffres manuscrits

Nous considérons également la base de données *Nist special database 3* qui consiste en des images bitmap 128 $\times$ 128 de chiffres et de lettres manuscrits. Nous nous concentrons sur un sous ensemble de chiffres, écrits par 100 scribes différents et pouvant

être tracés « en une ligne » (l'ensemble des pixels étant ainsi connexe). Chaque classe (chiffres de 0 à 9) a environ 1 000 instances, pour un corpus de 10 568 chiffres.

Comme nous travaillons sur des mots, chaque image est transformée en chaîne octale suivant un algorithme décrit par Micó & Oncina (1998) : à partir du pixel le plus en haut à gauche, on suit la frontière jusqu'à retomber sur le premier pixel. Durant son chemin, l'algorithme construit une chaîne incluant la direction du prochain pixel sur la frontière. La figure 2 donne un exemple de "2" ainsi codé.

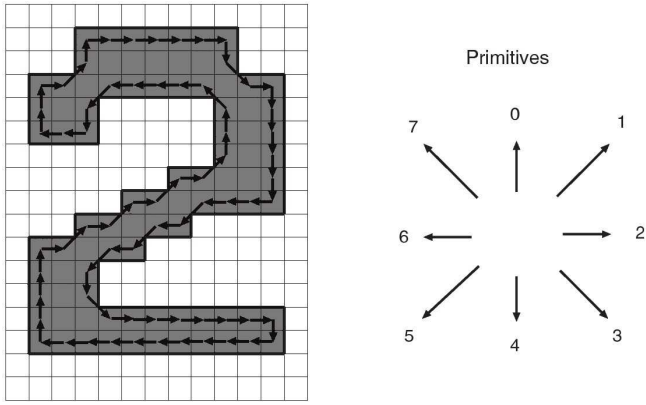

FIGURE 2 – Exemple de chiffre manuscrit. La chaîne équivalente vaut "2"=222222 432444446665656543222222246666666666000021212121007666654660021 0.

Pour ce corpus, nous nous comparons à Oncina & Sebban (2006) grâce à l'utilisation de SEDiL (Boyer *et al.* (2008)) et d'une matrice d'édition pondérée apprise sur ces données par Marc SEBBAN à l'aide d'un transducteur stochastique sur 8 000 paires de mots, une paire consistant en une chaîne et son plus proche voisin (soit environ 80% des données disponibles utilisés en apprentissage par SEDiL). L'étiquetage final consiste alors à attribuer la classe du plus proche voisin, les distances d'édition étant calculées grâce à la matrice d'édition pondérée apprise. Pour nos algorithmes, nous avons conservé notre protocole habituel, à savoir une validation croisée 10 fois, les apprentissages se faisant cette fois sur 10% des données, les tests sur les autres 90%. SEDiL a été évalué sur les mêmes exemples de test, ce qui indubitablement induit un biais en sa faveur puisque bon nombre des mots présents dans le test ont été vus en apprentissage par SEDiL. Les résultats sont donnés table 3.

3.3 Observations et bilan des expérimentations

Si l'on écarte la contre-performance de `centerCourt` sur le problème `tic-tac-toe`, on peut considérer que nos stratégies pour calculer le centre de la nouvelle boule sont très proches. Notons également qu'à de rares exceptions près, la qualité des prédictions augmentent avec le nombre de boules produites. Finalement, nos combinaisons de boules sont plus que compétitives en terme de prédiction vis-à-vis des autres méthodes testées, en particulier sur les données génomiques (problèmes `promoters` et

TABLE 3 – Précision sur Nist special database 3 pour 1 000, 10 000 et 100 000 boules produites par GLOBOOST (performance de SEDiL : 95.86 %).

	×1 000	×10 000	×100 000
G-B$_{\text{NATURELLE}}$	92.59 %	95.14 %	95.60 %
G-B$_{\text{GRAINE}}$	93.74 %	95.79 %	96.14 %
G-B$_{\text{CENTRELONG}}$	93.64 %	95.89 %	96.23 %
G-B$_{\text{CENTRECOURT}}$	92.92 %	95.73 %	96.15%
G-B$_{\text{HASARD}}$	**93.81 %**	**95.93 %**	**96.32 %**

`splice`) et sur la reconnaissance de caractères où nous dépassons SEDiL malgré notre protocole qui lui est très avantageux.

Le fait de pouvoir produire 100 000 hypothèses ou plus est caractéristique des boules de mots, cela n'est pas envisageable par exemple pour des automates 0-réversibles ou k-TSS. D'une part, les algorithmes de généralisation des automates sont trop longs pour fournir autant d'hypothèses rapidement et dans le cas de SEDiL c'est la classification elle-même qui nécessite un nombre quadratique de calculs de distance entre mots. D'autre part ce sont rapidement les mêmes automates qui sont produits. Autrement dit, les boules sont rapides à calculer et très diverses. Rappelons qu'une diversité importante est habituellement considérée comme un point fort pour les méthodes d'ensemble (Kuncheva & Whitaker, 2003).

Autre observation : pour chaque série d'expérimentations, les exemples se placent sur les bords des boules apprises, le centre n'est jamais un exemple du concept.

Exemple 9
Sur la base `tic-tac-toe` *qui regroupe les fins de partie au jeu du morpion, les parties gagnantes pour les croix constituant les exemples positifs : nous apprenons la boule de rayon 5 et de centre* `bbbb`*. Elle ne contient aucun négatif mais couvre 120 positifs, tous à distance 5 du centre :* `xxxoobbbb`, `xobxbbxbo`, `xbboxbobx`, `obxbbxobx`, `boxoxbxbb`, `bbxobxobx`, etc.

Exemple 10
Sur la base `us-first-name`*, base de prénoms américains parmi lesquels il s'agit de distinguer les prénoms féminins des masculins, la boule de centre* `LRLRTSVKCA` *et de rayon 7 est apprise : elle couvre 346 prénoms féminins et pas un masculin. À nouveau, les exemples couverts sont sur le bord de la boule.*

Bien que cela puisse s'expliquer, par notre méthode de construction des hypothèses et par les propriétés intrinsèques des boules de mots, cela reste néanmoins remarquable. En effet, lorsqu'elles sont utilisées dans le cadre d'un apprentissage en situations bruitées (comme par Tantini *et al.* (2006)), le centre est généralement une donnée non bruitée, et le rayon est considéré comme un degré de tolérance au bruit. Ici, le centre de l'hypothèse finale est alors plutôt à considérer comme une chaîne moyenne ou médiane des exemples positifs. Par ailleurs, nous pouvons noter que dans $\mathbb{R}^n$ une boule

avec les points de l'échantillon placés à la frontière serait indubitablement une boule moindre généralisée de ces points. Et pour terminer sur la diversité et sur la structure des boules, rappelons le résultat de VC-dimension infinie (théorème 7) : il éclaire d'une part la diversité des boules observées et sa preuve utilise elle aussi une boule creuse.

4 Conclusion

Nous avons traité dans cet article un problème classique, celui de l'exploration d'un espace d'hypothèses $\mathcal{H}$ structuré par une relation de généralité $\succeq$. Les stratégies de parcours sont nombreuses et difficiles à fonder. Nous pensons que, quand il existe, c'est l'opérateur qui calcule un *moindre généralisé unique* qui doit être utilisé. Cependant, nous avons montré qu'en son absence un certain esprit de la méthode pouvait être conservé, ainsi que certaines de ses propriétés.

Nous avons appliqué cette démarche à l'apprentissage de boules de mots dont nous avons montré qu'elles permettent des moindres généralisés multiples. Une piste possible était de restreindre la classe pour revenir à un moindre généralisé unique mais il nous semblait regrettable de renoncer à la classe des boules. Nous aurions pu également envisager de maintenir plusieurs hypothèses en même temps à la manière d'une *beam search*, ce qui amenait alors les questions de sa taille, de la complexité induite et du choix des hypothèses à conserver. Finalement, nous avons choisi de ne travailler qu'avec une seule hypothèse, choisie dans la classe générale des boules de mots. Nous avons donc envisagé de suivre l'un des moindres généralisés et naturellement nous avons considéré celui qui correspond à la boule de plus petit rayon, mais nous avons vu que le calcul d'une telle boule ne pouvait être qu'exponentiel. Finalement, nous avons opté pour une boule plus générale que certains moindres généralisés mais encore raisonnablement proche des exemples.

Les résultats expérimentaux semblent valider notre démarche : les combinaisons de boules sont meilleures que les combinaisons d'automates sur des problèmes classiques de classification de séquences et elles sont meilleures que la méthode de référence sur un problème de reconnaissance de caractères. De plus, l'apprentissage est rapide car quand on travaille sur les boules, on ne fait que des opérations sur les mots, au contraire des automates pour lesquels les opérations sont plus complexes (fusion d'états, déterminisation, *etc.*). Ces bons résultats peuvent sembler étonnants au regard de la relative pauvreté d'une boule qui ne dénote qu'un langage fini mais nous avons relativisé cette apparente pauvreté par la VC-dimension infinie des boules de mots.

Au final, nous avons été capables d'intégrer des hypothèses à moindres généralisés multiples dans le système VOLATA, dédié au moindre généralisé unique. Cela ouvre le champ des classes d'hypothèses pouvant intégrer ce système et encourage à l'explorer. Parmi les autres perspectives, nous pensons que nos méthodes peuvent s'améliorer si l'on utilise des coûts différents sur les opérations d'édition, coûts appris spécifiquement pour chaque problème. Enfin, les bons résultats obtenus sur les données génomiques nous poussent à expérimenter un peu plus dans ce domaine.

Remerciements

Merci à Jean-Christophe JANODET pour le résultat sur le VC-dimension des boules de mots et à Marc SEBBAN pour les échanges sur la reconnaissance de caractères et SEDiL.

Références

BLAKE C. & MERZ C. (1998). UCI repository of machine learning databases [http ://archive.ics.uci.edu/ml/].

BOYER L., ESPOSITO Y., HABRARD A., ONCINA J. & SEBBAN J. (2008). Sedil : Software for edit distance learning. In W. DAELEMANS, B. GOETHALS & K. MORIK, Eds., *Proceedings of the 19th European Conference on Machine Learning*, p. 672–677 : Springer.

DE LA HIGUERA C. & CASACUBERTA F. (2000). Topology of strings : median string is NP-complete. *Theoretical Computer Science*, **230**, 39–48.

GANASCIA J.-G. (1993). Algebraic structure of some learning systems. In *ALT '93 : Proceedings of the 4th International Workshop on Algorithmic Learning Theory*, p. 398–409, London, UK : Springer-Verlag.

JANODET J.-C. (2010). Preuve de vc-dimension infinie pour les boules de mots. Communication personnelle.

KUNCHEVA L. I. & WHITAKER C. J. (2003). Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, **51**(2), 181–207.

LEVENSHTEIN V. I. (1965). Binary codes capable of correcting deletions, insertions, and reversals. *Doklady Akademii Nauk SSSR*, **163**(4), 845–848.

MICÓ L. & ONCINA J. (1998). Comparison of fast nearest neighbour classifiers for handwritten character recognition. *Pattern Recognition Letter*, **19**(3-4), 351–356.

ONCINA J. & SEBBAN M. (2006). Learning stochastic edit distance : Application in handwritten character recognition. *Pattern Recognition*, **39**(9), 1575–1587.

TANTINI F. (2009). *Inférence grammaticale en situations bruitées*. PhD thesis, Université Jean Monnet de Saint-Étienne.

TANTINI F., DE LA HIGUERA C. & JANODET J.-C. (2006). Identification in the limit of systematic-noisy languages. In *ICGI*, p. 19–31.

TORRE F. (2004). GloBoost : Boosting de moindres généralisés. In M. LIQUIÈRE & M. SEBBAN, Eds., *Actes de la Sixième Conférence Apprentissage CAp'2004*, p. 49–64 : Presses Universitaires de Grenoble.

TORRE F. & TERLUTTE A. (2009). Méthodes d'ensemble en inférence grammaticale : une approche à base de moindres généralisés. In Y. BENNANI & C. ROUVEIROL, Eds., *11ème Conférence francophone sur l'Apprentissage automatique (CAp'2009)*, p. 33–48, Hammamet (Tunisie) : PUG.

WAGNER R. A. & FISCHER M. J. (1974). The string-to-string correction problem. *Journal of the ACM*, **21**, 168–178.